APPLICATION

FOR

UNITED STATES LETTERS PATENT

TITLE: CRYPTOGRAPHIC POLICY FILTERS AND POLICY
             CONTROL METHOD AND APPARATUS

APPLICANT: TAHER ELGAMAL, JEFF WEINSTEIN AND NELSON
             BOLYARD

CRYPTOGRAPHIC POLICY FILTERS AND POLICY CONTROL

METHOD AND APPARATUS

BACKGROUND OF THE INVENTION

5   1.   Field of the Invention

The present invention relates to cryptography configuration. More particularly, the present invention relates to method and apparatus for controlling the use of cryptography such that products utilizing these controls may be exported in accordance with United States export laws, and/or imported into other countries that

10   place additional restrictions on the use of cryptography.

2.   Description of the Related Art

There are many circumstances where the distribution or the use of encryption software is regulated by governments. In some countries, the strength of encryption

15   that can be exported is regulated without imposing any restrictions upon the distribution of the encryption software within the country. For example, in the United States, companies are free to distribute any type of encryption software developed within the country for use by United States citizens. Furthermore, United States allows unrestricted importation of encryption technology. However, exporting of a

20   certain strength encryption in the United States is regulated. In other countries, such as France, the strength of encryption that can be used, distributed or imported is tightly regulated.

In the case where the exporting of the encryption software is restricted, the permissible exportable encryption software are usually limited to specific algorithms

25   that use key sizes which are weaker than a particular size. Previously, the encryption software has generally been an integral part of a software application. Therefore, to accommodate the varying degrees of allowed encryption levels, several versions of the same application are typically created; one version that provides strong encryption by those who are allowed unrestricted use, and one or more versions that use weaker

30   encryption for those customers whose use is restricted. In many cases, one single weaker version is created to avoid the expense of creating and maintaining multiple

1

versions of the same application.

One reason that applications distributed under these restricted rules have been required to include encryption modules as an integral part of the software, rather than as a separate module that can be plugged into the application, is to prevent an

5   encryption module that supports strong encryption from being plugged into an application and used in a country where encryption is restricted, and thereby enabling strong encryption with that application.

Exportable/export strength encryption refers to encryption algorithms and key sizes that provide relatively weak (i.e., easy to defeat) protection of information.

10   Presently, the United States government allows general purpose encryption products to be exported from the United States to other countries only if those products utilize approved encryption algorithms and key sizes that are weaker than a certain measure. General purpose encryption is an application of encryption that does not limit the type or size of the data being encrypted. The United States government places more

15   restrictions on software that can encrypt any kind of data than it does on software that limits the type and size of the data being encrypted. SSL and S/MIME are considered to be general purpose encryption protocols because they do not place any limits on the type or size of the data being encrypted.

By contrast, special purpose encryption is an encryption application that limits

20   the type and size of the data being encrypted. The United States government will sometimes give export permission for stronger encryption algorithms if they are limited in use to a specific purpose and type of data (strong encryption defined as encryption algorithms and key sizes that are stronger (i.e., harder to defeat) than those that the United States government usually allows to be exported for general purpose

25   encryption). The SET protocol is an example of an encryption protocol that was designed to encrypt special purpose data for purposes of performing financial transactions only.

Recently, standards have been developed that define the interfaces between application software and modules that provide encryption. The purpose of these

30   standards is to facilitate the use of Smart Cards and other hardware encryption devices. By defining a standard interface, and allowing encryption modules to be plugged into applications, any encryption module can be used with any application without having to write special software to integrate each module with each application.

2

The advent of these new standards presents a problem for application developers since applications that support pluggable encryption modules will be difficult or impossible to distribute under the current varying government regulations. In order to obtain the benefits of standards for pluggable encryption modules, an

5   approach is necessary that satisfies governments by way of ensuring that their various levels of restrictions will be enforced. Such an approach will allow the application to be distributed separately from any encryption modules that it may use, but still allow the application to restrict the use of encryption to the permissible level.

One approach has been developed by Microsoft for their Crypto API (CAPI)

10   system. Applications that use CAPI may only load encryption plugin modules that have been digitally signed by Microsoft. Microsoft will only sign a module once the developer of the module has agreed to abide by the governmental restrictions when distributing the module.

However, this approach poses several new problems. First, it requires a legal

15   agreement between Microsoft and all developers of encryption modules, which can be expensive and time consuming. Another problem with the CAPI system is that it does not allow the application to access stronger encryption algorithms that may otherwise be available in the encryption plugin for those cases where only special purpose encryption is being performed for higher level of encryption transactions, and stronger

20   encryption may be used legally. Some applications that would benefit from or require this type of access are: Secured Socket layer (which are exportable cipher suites that have 40-bit secret keys still use 128-bit RC2 or RC4 encryption, where the 40 bits of the 128-bit key are secret while the remaining 88 bits are not.); Special purpose encryption protocols such as SET (these protocols limit the scope and the size of the

25   data that they encrypt, so that most governments allow them to use much stronger encryption algorithms and key sizes than would be allowed for general purpose encryption); and encryption of key material (generally, when key material is being encrypted, strong algorithms and larger key sizes can be legally used).

As previously discussed, because the level of permissible exportable

30   cryptography varies depending upon the importing country, different versions of a single application are necessary. Also, these different versions of the same application differ in languages used in the manuals and in the help files that are part of the product. In the past, a separate version of the single application was developed for each market, embodying the legally marketable cryptographic policies, for sale and

3

use of the application in the target market. Each of these different versions had to be separately built, tested, localized (translated into the local languages), and supported. The cost of these operations is considerably greater than if there was only one single version of the application to serve all the markets. Having a single application for all

5   these markets can also reduce development and testing time, therefore, shorting the time between development and testing of a new application to marketing and sale.

As discussed above, to address the multiple-version issue, some manufacturers and developers only make and sell products that contain no cryptography, while others only make and sell products that contain only the weaker exportable forms of

10   cryptography, the latter foregoing making stronger cryptography available inside the United States (where stronger cryptography is not subject to export regulations). There are also manufacturers and developers that make separate version of the same program for each of the differing markets, and bear the expense and time-to-market cost associated with making and supporting multiple versions of the same

15   application/product.

With any of the aforementioned approaches, many customers including banks and financial institutions find the weaker level of security offered as exportable general purpose encryption to be unacceptable for financial transactions. Additionally, building separate programs substantially increases time and expenses in production,

20   quality testing, and support for the products.

SUMMARY OF THE INVENTION

In view of the foregoing, in accordance with the present invention, policy filters are provided in which an application is limited to accessing only those

25   algorithms in an encryption module that are indicated in the policy filters, even if stronger encryption algorithms are available in the encryption module. The policy filter in accordance with the present invention allows anyone to develop a cryptographic plugin module without the need to have legal agreements between the application or API developers and the crypto module developers.

30   Moreover, the policy filters in accordance with the present invention allow separate controls to be used for the different APIs, Protocols, and Services in a system, so that the unique needs of each market can be met. The policy filters in accordance with the present invention are provided as a part of a system that includes

4

cryptographic functionality. The cryptographic functionality includes information indicative of the cryptographic policies such as the various levels of allowable encryption between the different countries. The system including cryptographic functionality could be an operating system or an application platform such as the

5   Netscape ONE platform as it exists in Netscape Communicator® or SuiteSpot® servers. These systems have cryptographic APIs, Protocols, and Services that are made available to layered applications. For further background information on layered applications, see pending U.S. Patent Application S/N 08/519,585.

Moreover, in accordance with another embodiment of the present invention,

10   there is provided a cryptographic policy file including a cryptographic policy module. The cryptographic policy module, which is part of an application, accesses the cryptographic policy file when the application is executed and performs several validity verifications on the cryptographic policy file to ensure that the cryptographic policy file is legitimate and is being used in the market for which it was intended. If

15   the cryptographic policy module determines that the cryptographic policy file passes these tests, the cryptographic policy module then enables or disables each of the possible forms of encryption which the application can request according to the contents of the cryptographic policy file.

Each of the program's cryptographic applications, for example, SSL,

20   S/MIME, and PKCS #12, provides a uniform interface for turning each of its capabilities on or off, and for setting numeric parameters (e.g., key sizes). Also, each of these capabilities is pre-configured to a default "off" or "minimal security" state, and remains in that state until re-configured by the cryptographic policy module according to the contents of the cryptographic policy file when the application is first

25   executed. Accordingly, a separate cryptographic policy file is provided for each market. Therefore, a single program can legally be used and sold in each of the aforementioned markets with varying levels of supportable encryption, and will provide the forms of encryption that are legally appropriate to each market according to the cryptographic policy file supplied to the specific market.

30   These and other features and advantages of the present invention will be understood upon consideration of the following detailed description of the invention and the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 illustrates a block diagram of a system including a policy filter in accordance with one embodiment of the present invention.

Figure 2 illustrates a flow chart of the initialization of the policy filter in accordance with one embodiment of the present invention.

Figure 3 illustrates a flow chart of the control of capability query through the policy filter in accordance with one embodiment of the present invention.

Figure 4 illustrates the control of cryptographic operation through the policy filter in accordance with one embodiment of the present invention.

Figure 5 illustrates a flow chart of a cryptographic policy module using a cryptographic policy file in accordance with one embodiment of the present invention.

Figure 6 illustrates a block diagram of a system including a policy file and module in accordance with one embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

Figure 1 illustrates a block diagram of a system including policy filters in accordance with one embodiment of the present invention. Various application programs-- Messenger 101, the Navigator 102, and a "Third Party" Application 103 are shown. A plurality of policy filters are provided including a crypto API policy filter 108, an SSL policy filter 109, an S/MIME policy filter 110, and a key storage policy filter 111. Each of these policy filters is associated with a corresponding service module including a crypto API 104, an SSL 105, an S/MIME 106, and a key storage service 107, respectively.

Referring to Figure 1, there is also provided a policy filter initialization module 112 which configures each of the policy filters (including the crypto API policy filter 108, the SSL policy filter 109, the S/MIME policy filter 110, and the key storage policy filter 111) of the system prior to the execution of the first cryptographic operation. There are also provided three cryptographic plugin modules including Fortezza card 114, cryptographic smart card 115, and a software crypto module 116.

Public key cryptography standard number 11 (PKCS #11) 113 operates as a

standard interface to crypto modules. A crypto module, a cryptographic module, or an encryption module is a software module that provides a standard interface to low level cryptographic services such as encryption and decryption of data.

5   As will be further explained in more detail below in conjunction with Figure 2, when one of the applications (the Messenger 101, the Navigator 102, or the third party application 103) is executed, the crypto plugin modules 114, 115, and 116 are loaded and the policy filters 108, 109, 110 and 111 are configured by the policy filter initialization module 112. Moreover, the policy filter initialization module 112 performs conformance tests upon the loaded crypto modules 114, 115, and 116.

10   Having thus initialized the policy filters 108-111 in the manner described above, service module capabilities are next determined as will be explained in more detail in conjunction with Figure 3. Thereafter, when an application program (i.e., the Messenger 101, the Navigator 102, or a third party application 103) calls a service module (i.e., the crypto API 104, the SSL 105, the S/MIME 106, or the key storage

15   service 107) to perform an operation that involves cryptographic functions, the called service module calls its respective policy filter (i.e., the crypto API policy filter 108, the SSL policy filter 107, the S/MIME policy filter 110, or the key storage policy filter 111) to determine whether the called operation is allowed. If the called operation is not allowed, then the service module returns an error to the application. If, however,

20   the called operation is allowed, the service module performs the called operation, calling the cryptographic module when necessary, and returning the results to the application.

Figure 2 is a flow chart illustrating the initialization of the policy filters in accordance with one embodiment of the present invention. As shown, when the

25   system of Figure 1 begins the initialization at step 202, the system first loads the crypto modules (for example, the Fortezza card 114, the cryptographic smart card 115 and the software crypto module 116) at step 203. Loading the crypto modules entails invoking a dynamic mechanism provided in operating systems for loading modules of code, referred to as shared libraries or dynamic link libraries (DLLs), into an already

30   running application program. In this manner, the crypto module code is dynamically loaded by the operating system into the running application program. The system then configures the policy filters at step 204.

The policy filter initialization module 112 (Figure 1) then performs conformance tests on the crypto modules at step 205. The conformance tests involve

7

requesting the crypto module to perform a cryptographic function, then comparing the results to the result of a known compliant implementation of the same algorithm. If, at step 206, the conformance tests performed by the policy filter initialization interface 112 is successful, then a result of success is returned to the system at step 208. If,

5   however, the conformance test performed at step 206 is not successful, (i.e., the crypto module does not correctly implement the algorithms and/or key sizes configured), then a result of failure is returned to the system at step 207, followed by a disabling of the crypto module at step 209 which failed the conformance test.

In the manner described above, the initialization of the policy filter prevents a

10   cryptographic module from incorrectly stating its capabilities in order to bypass the system's controls. Accordingly, the conformance tests in accordance with the present invention are sufficiently broad to ensure that the cryptographic module is correctly implementing the algorithms and that the key sizes advertised therefrom are indeed being used.

15   Figure 3 is a flow chart illustrating the control of capability query through a policy filter (i.e., one of the policy filters 108-111 of Figure 1) in accordance with one embodiment of the present invention. As shown, when the query begins at step 301, the application initially queries the capabilities (i.e., cipher suites that it supports) of the service modules at step 302. Accordingly, at step 303, a corresponding service

20   module (which is one of the service modules 104-107 of Figure 1) generates a full list of cipher suites that it supports and passes this list to the policy filter. In this regard, a cipher suite is a collection of encryption algorithms, key sizes, and parameters that fully specifies the type and the strength of a particular cryptographic operation.

At step 304, the policy filter applies its configuration parameters to filter out

25   or remove unauthorized cipher suites from the list received from the service module at step 303. After filtering at step 304, the policy filter then returns the filtered list of cipher suites to the service module at step 305, which, in turn, returns the filtered list of cipher suites to the application. With the list of filtered (or authorized) cipher suites, the application causes a cryptographic operation to be performed in accordance

30   with Figure 4.

Referring now to Figure 4, the cryptographic operation is initiated by an application at step 401. The application calls a service module to request an operation involving cryptographic functions at step 402. At step 403, the service module calls its corresponding policy filter to determine whether the called operation is allowed. At

step 404, if the called operation is not approved by the corresponding policy filter, then the service module returns an error to the application at step 405. On the other hand, if, at step 404, the called operation is approved, then at step 406, the service module performs the called operation, calling the cryptographic module as necessary. Thereafter, the service module, at step 407, returns the operation results to the application.

In accordance with the present invention, the policy filters are configured by the system as shown in Figure 1 before the first cryptographic operation is performed. The policy filter module provides a configuration interface for this purpose. The policy filters will have a default configuration, which may be to disable all operations, or to allow a subset of operations that are generally allowed by the most restrictive configurations. There are numerous mechanisms that the system may use to determine the correct configuration parameters for the policy filters. Some examples include, but are not limited to, compiling the parameters into the system or reading the parameters from a protected file (e.g., such as a digitally signed file). It is to be noted that some application programs may be compiled into one system, while other application programs may by dynamically loaded.

The policy filter module also performs conformance tests on the cryptographic module to ensure that the crypto module is correctly implementing the algorithms and key sizes that it is advertising. This is to prevent a cryptographic module from incorrectly stating its capabilities in order to bypass the system's controls. The sets of tests are broad enough to ensure that the module is correctly implementing the algorithms and that the key sizes being advertised are indeed being used.

As illustrated above, the policy filters in accordance with the present invention are associated with each service module that will use the cryptographic services provided by the cryptographic module. The policy filters control both the capabilities advertised by the service modules and the operations the service modules are allowed to perform. The process of requesting the capabilities of a service module involves first the application querying the service module for the list of cipher suites that it supports. In response, the service module passes its list of supported cipher suites to its policy filter. With the list of supported suites, the policy filter applies its configuration parameters to filter the list of cipher suites. Having filtered the list of cipher suites, the policy filter returns the filtered list of cipher suites to the service module, which, then returns the filtered list of cipher suites to the application.

In accordance with another embodiment of the present invention, there are provided cryptographic policy files comprising a set of ASCII-coded Attribute-Value pairs which may optionally be compressed, and a digital signature on the set of Attribute-Value pairs. The set of Attribute-Value pairs is a series of lines of plain ASCII text. Each line contains the name of an attribute of the cryptographic policy, followed by a colon, optionally some space, and the value of that attribute. A value may be in the form of a sequence (or "string") of printable ASCII characters, an integer number, or a "truth expression", that is, one of the words "true", "false" or "conditional." The following Table 1 illustrates the attribute name, its type and purpose, all of which are required in each cryptographic policy file.

TABLE 1 Attribute in Cryptographic Policy Files

Attribute Name	Type	Purpose
POLICY-BEGIN-HERE	String	Name the policy (market) for this file, for example, "Export Policy."
Software-Version	String	Identifies the program, its version number, and the localization (language) incorporated in the program, for example, "Mozilla/4.0P3 [fr]."

All other attributes are dependent upon the cryptographic capabilities of the program. Each and every cryptographic capability of the program that is subject to the export laws of the United States, or to the import laws of other countries, must be named as an attribute of the cryptographic policy in each of the cryptographic policy files produced for that program. The following Table 2 illustrates a sample Attribute-Value pairs incorporated into a program.

TABLE 2 Sample Attribute-Value Pairs

POLICY-BEGINS-HERE:	Export policy
Software-Version:	Mozilla/4.0P3 [fr]
MAX-GEN-KEY-BITS:	512
PKCS12-DES-EDE3:	false
PKCS12-RC2-128:	false
PKCS12-RC4-128:	false
PKCS12-DES-56:	false
PKCS12-RC2-40:	true
PKCS12-RC4-40:	true
PKCS12-NULL:	true
SSL2-RC4-128-WITH-MD5:	false
SSL2-RC2-128-CBC-WITH-MD5:	false
SSL2-DES-168-EDE3-CBC-WITH-MD5:	false
SSL2-DES-56-CBC-WITH-MD5:	false
SSL2-RC4-128-EXPORT40-WITH-MD5:	true
SSL2-RC2-128-CBC-EXPORT40-WITH-MD5:	true
SSL3-FORTEZZA-DMS-WITH-FORTEZZA-CBC-SHA:	false
SSL3-FORTEZZA-DMS-WITH-RC4-128-SHA:	false
SSL3-RSA-WITH-RC4-128-MD5:	conditional
SSL3-RSA-WITH-3DES-EDE-CBC-SHA:	conditional

SSL3-RSA-WITH-DES-CBC-SHA:	false
SSL3-RSA-WITH-RC4-40-MD5:	true
SSL3-RSA-WITH-RC2-CBC-40-MD5:	true
SSL3-FORTEZZA-DMS-WITH-NULL-SHA:	false
SSL3-RSA-WITH-NULL-MD5:	true
SMIME-DES-EDE3:	false
SMIME-RC2-CBC-128:	false
SMIME-RC5PAD-64-16-128:	false
SMIME-DES-CBC:	false
SMIME-RC2-CBC-64:	false
SMIME-RC5PAD-64-16-64:	false
SMIME-RC2-CBC-40:	true
SMIME-RC5PAD-64-16-40:	true

The set of attribute-value pairs in the cryptographic policy file in accordance with the present invention can also be compressed to reduce its size and to make the contents less apparent to others.

5   As previously explained, the second component of the cryptographic policy file is the digital signature. A digital signature is a block of data (a very large number) computed mathematically from another set of data (the "signed data") using the signer's "private key". The digital signature may be used to verify that the signed data has not been altered in any way since it was signed, and that it was genuinely signed

10   by the party named in the signer's certificate. The digital signature affirms the authenticity and the integrity of the signed data, which, in accordance with the present invention, is the string of Attribute-Value pairs.

12

The digital signature is stored in the file as a part of a "signed-data content type" block, as defined in Public Key Cryptography Standard #7 (PKCS #7). A signed-data content type block is a block of data in the format standardized for the representation of digital signatures. It contains a digital signature, the certificate used

5 to verify the signature (the "signer's certificate"), the chain of Certification Authority (CA) certificates used to validate the signer's certificate, and optionally the data that was signed by the digital signature (the "signed data"). An "external Signature" is a signed-data content type data block containing a digital signature, the signer's certificate, and a certificate chain, but not containing the signed data.

10 A Certification Authority is an entity that issues (or signs) digital certificates. Furthermore, a Certificate Chain is a list of certificates, each one (except the last) issued, signed, and authorized by the party identified in the succeeding certificate in the list. The Certificate Chain is used to determine the validity and veracity of the first certificate in the chain, which is usually the certificate of a document's signer. The

15 final certificate in the chain must be that of a trusted party for the chain to have validity.

Accordingly, the Attribute-Value pairs and the External signature are stored in an archive file known as a "jar" file. The jar file provides a convenient mechanism in which to store multiple components together in one file. Moreover, in accordance

20 with the present invention, the jar-file utility programs provide a convenient method to compress the selected components, and to sign the components. It is to be noted, however, in accordance with another embodiment of the present invention, the Attribute-Value pairs are stored within the PKCS #7 signed data content type block, in which case the PKCS#7 signed data content type block is not said to be an External

25 signature, and the policy file consists solely of the PKCS#7 signed data content type block.

A Private Key is a block of data (actually a very large number) used to sign (make a digital signature for) another document. The private key must be kept secret by the signer if the signatures it generates are to be trusted. On the other hand, a

30 public key is a block of data (actually a very large number) used to verify the digital signature on a digitally-signed document. Unlike the Private Key, the Public Key need not be kept secret, and may be distributed to the public. A Public Key is usually conveyed in a signed document called a certificate. A certificate is a document containing a signer's name and public key, and the signature of a third party, vouching

for the accurate identification of the party named in the certificate. A certificate is used to obtain the public key needed to verify a digital signature made by the party named in the certificate.

5 Furthermore, as illustrated above, in accordance with the present invention, the cryptographic policy module in the program performs its services when the user begins the program's execution on the user's computer. It also performs its services whenever the user attempts to change the configuration of the cryptographic functions within the program.

10 As shown in Figure 5, when the program begins executing, the cryptographic policy module first reads the cryptographic policy file into the computer's memory (as shown in step 501). Then, the policy module verifies the digital signature on the Attribute-Value pairs at step 502. At step 503, the module verifies that the digital signature is a genuine digital signature of the product's manufacturer by determining that one or more of the certificates in the digital signature's certificate chain is the certificate issued by the manufacturer for that very purpose. This is accomplished in

15 the present invention by comparing to a "built-in" CA certificate. At step 504, a further verification is performed which confirms that the Software-Version Attribute value string matches the version of the running program. Finally, at step 505 yet another verification is performed, confirming that the localization (language)

20 specified in the Software-Version Attribute value string matches the localization in use by the running program.

If the verification in any of the above steps 502-505 fails, then the entire content of the cryptographic policy file is ignored at step 506, and the program behaves according to its default set of encryption rules, which generally is the most

25 restrictive set of cryptographic capabilities in the program. If, on the other hand, all of the verification steps 502-505 are successful, then the cryptographic policy module proceeds to step 507 where the cryptographic software in the running application program is configured for each and every one of the cryptographic capabilities listed in a set of Attribute-Value pairs. The capabilities that are enabled or disabled

30 according to a "truth expression" are enabled or disabled according to the expression in the relevant Attribute-Value pair in the corresponding policy file. The capabilities that vary by number (for example, key length) are set to the numeric value in the relevant pair. Any values not specified are left in their default configuration, which is generally off, or using the smallest numeric value.

14

Figure 6 illustrates a block diagram of a system including a policy file and a cryptographic policy control module in accordance with one embodiment of the present invention. In such a system, there is provided an application program such as the Netscape Communicator® 601 which includes a plurality of applications such as

5 the Navigator SSL 602, the Messenger S/MIME 603, and the Key Storage PKCS #12 603 as three cryptographic applications. There is also provided a cryptographic policy control module 605 which configures each of the three cryptographic applications, Navigator SSL 602, the Messenger S/MIME 603, and the Key Storage PKCS #12 603 in accordance with a predetermined data string received from a policy file 606. Policy

10 files can be used to directly configure a trusted application, or can be used as the mechanism to configure policy filters as described in the first embodiment of the present invention.

Also, in accordance with the present invention, the application's user is generally allowed to select preferences among the various cryptographic capabilities

15 available, and may elect to disallow certain of the approved capabilities, or to re-enable capabilities that he has previously disallowed. In this manner, the user may alter those preferences while the program is running, When the user attempts to do so, the cryptographic policy module is again used to ensure that the user cannot enable any capabilities that are disallowed by the cryptographic policy file.

20 In this manner, according to the present invention, a single program can be made and sold in all markets. The manufacturer spends only as much production, testing and support as a single program requires, and the program provides stronger encryption where it is legally allowed.

Various other modifications and alterations in the structure and method of

25 operation of this invention will be apparent to those skilled in the art without departing from the scope and spirit of the invention. Although the invention has been described in connection with specific preferred embodiments, it should be understood that the invention as claimed should not be unduly limited to such specific embodiments. It is intended that the following claims define the scope of the present

30 invention and that structures and methods within the scope of these claims and their equivalents be covered thereby.